Autorun for Integrated Circuit Memory Component

Technical Field

[0001] This invention relates to a system and method for utilizing storage media such as flash memory for achieving autorun of an application executable or installer stored on the storage media.

Background and Summary of the Invention

[0002] As is known in the art, some applications such as software installers may be run automatically upon insertion of a CD-ROM disc into a CD-ROM drive, which may sometimes be called a dock or reader. In operation, this automatic running of an application is provided by an autorun feature that is stored on or incorporated into CD-ROM drive dock/reader. Executables or installers stored on the CD-ROM disc are executed by the host personal computer based upon activation by the autorun feature in the CD-ROM drive dock/reader. In this implementation, the autorun feature is incorporated into the hardware drive/dock/reader, which is separate from the storage media.

[0003] Universal Serial Bus (USB) technology is rapidly gaining preference as the interfacing technology of choice for peripherals on computing devices such as personal or laptop computers. Flash memories coupled with a USB interface has become a convenient and portable storage device that can replaces floppy disks and compact disks (CDs).

[0004] However, the popular and widely-adopted Universal Serial Bus technology does not include distinct autorun features in the docks/readers. As a consequence, conventional integrated circuit memory devices such as USB memory devices do not have autorun functionality.

[0005] Accordingly, the present invention provides autorun functionality to any IC memory device, such as any USB peripheral, that has a memory component interfaced to a computing device interface microcontroller. The present invention provides autorun of one or more executables or application installers from a memory component

with an interface to a computing device without an intermediate hardware-based autorun feature. As an example, such interface could be a USB interface and such computing device could be a personal computer.

[0006] For example, each USB peripheral device internally contains a USB microcontroller that performs the functionality associated with identifying the device to a host computing device, such as a personal computer. In accordance with the present invention, autorun firmware is embedded into the USB microcontroller. The autorun firmware enables autorun of an installable or executable application stored on the memory component of the USB device. The firmware acts as bridge component translating all commands and interactions between a host PC and the memory component.

[0007] Additional description and implementations of the present invention will be apparent from the detailed description of the preferred embodiment thereof, which proceeds with reference to the accompanying drawings.

Brief descriptions of the Drawings

[0008] Fig. 1 illustrates an exemplary implementation of an autorun integrated circuit (IC) memory device according to the present invention.

[0009] Fig. 2 is a block diagram of a prior art arrangement in which a host personal computer includes an intermediate hardware dock that provides an autorun feature.

[0010] Fig. 3 is a flow diagram of an IC memory device autorun method.

[0011] Figs. 4A and 4B illustrate autorun firmware according to the present invention be embedded into alternative USB device configurations

[0012] Fig. 5 is a block diagram of a USB peripheral having multiple functionalities.

[0013] Fig. 6 is a block diagram of a USB hub with autorun firmware and access to multiple distinct functionalities.

[0014] Fig. 7 is a schematic diagram of a person-operable physical slide switch.

[0015] Fig. 8 is a flow diagram of a software-implemented copy protection method.

Detailed Description of Preferred Embodiment

[0016] Fig. 1 illustrates an exemplary implementation of an autorun integrated circuit (IC) memory device 100 according to the present invention. Autorun IC memory device may be in the form of a USB memory device, a compact flash card, a smart card, etc. For purposes of illustration, autorun IC memory device 100 will be described with reference to a universal serial bus (USB) memory device implementation.

[0017] Autorun IC memory device 100 includes a memory component 110 that communicates with a USB microcontroller 120 having autorun firmware 130 incorporated or embedded into microcontroller 120. Autorun IC memory device 100 includes an upstream port 140 for connecting to a host computing device 150 (e.g., personal or laptop computer, handheld computer, PDA, smart phone, etc., not shown). In the illustrated implementation, upstream port 140 is a USB port.

[0018] Autorun firmware 130 causes an application or executable stored in memory component 110 to be installed or run automatically upon activation of the IC memory device 100 vis-à-vis the host computing device 150. This activation may be achieved in a variety of ways including connecting or inserting the autorun IC memory device 100 into a docking system or port present on or interfaced to the host computing device 150. For example, IC memory device 100 with autorun firmware 130 incorporated into USB microcontroller 120 allows a "USB Flash Drive" storing one or more application executables or installables to be run automatically (i.e., autorun) upon activation, such as being plugged into the USB port of a host PC 150.

[0019] Fig. 2 is a block diagram of a prior art arrangement in which a host personal computer 200 includes an intermediate hardware dock 220 that provides an autorun feature for a storage medium like a CD-ROM 230. Intermediate hardware dock 220 functions as a storage media

reader that may be internally integrated with or externally connected to the host personal computer 200 and the storage medium 230.

[0020] In this prior art implementation, insertion of a CD-ROM disc 230 into a CD-ROM dock/reader 220 may cause activation of an autorun feature that is stored on or incorporated into CD-ROM dock/reader 220. Executables or installers stored on the CD-ROM disc 230 may then be executed by the host personal computer 200 based upon activation by the autorun feature CD-ROM dock/reader 220.

[0021] As another example of such a prior art implementation, a flash memory card reader connected to a host computing device, such as a personal computer, may also include an autorun feature that can activate an executable or installer to run on the host computing device.

[0022] A disadvantage of such prior art implementations is that autorun features are incorporated into hardware docks or readers that are separate from the storage media. However, the popular and widely-adopted Universal Serial Bus technology does not include such distinct autorun features. As a consequence, conventional integrated circuit memory devices such as USB memory devices do not have autorun functionality. In contrast, the present invention provides autorun functionality to any IC memory device, such as any USB peripheral that has a memory component interfaced to a USB microcontroller.

[0023] Fig. 3 is a flow diagram of an IC memory device autorun method 300 that may be implemented from firmware 130 incorporated into a USB controller 120.

[0024] In step 305, a USB peripheral is inserted into or connected to a USB port of a host computing device (e.g., a personal computer).

[0025] In step 310, the host computing device performs an enumeration to identify the newly attached USB peripheral.

[0026] Step 320 is a query as to whether the USB peripheral includes autorun firmware that is enabled. If so, step 320 proceeds to step 330. If not, step 320 proceeds to step 370.

[0027] In step 330, the autorun firmware in the USB peripheral announces itself with a device interface description. For example, the device interface description may include Mass Storage Class, SCSI transparent command set, Bulk Only Transport corresponding to a CD-ROM, for example.

[0028] In step 340, the host and the USB peripheral communicate with each other using, for example a standard MMC-2 specification set. The communication includes a response to host commands from the autorun firmware according to the MMC-2 specification. As a part of the MMC-2 specification, the host requests enumeration of files in root directory and the autorun firmware responds to the request.

[0029] In step 350, the autorun firmware informs the host of the presence of an autorun executable file to be executed and provides the file to the host. For example, the file may be named "Autorun.inf," which may be stored on the memory component of the USB peripheral. The host executes the autorun executable file to provide the autorun functionality.

[0030] Step 360 is a query whether the autorun firmware is to be enumerated again or "re-enumerated." If so, step 360 proceeds to step 370. If not, step 360 proceeds to step 390. Re-enumeration allows the autorun firmware to announce itself to the host as one or more other USB peripherals (e.g. data storage device, communication adapter, etc.) or, if there is no re-enumeration, the autorun firmware can continue to function as per MMC-2 specifications.

[0031] In step 370, the autorun firmware re-enumerates or identifies itself as another USB device, such as a USB flash drive or a USB wireless (e.g., Bluetooth, WiFi, IrDA) device or "dongle." With such a re-enumeration, the autorun firmware identifies itself with device interface descriptors for the other USB devices (e.g., USB flash drive or USB Bluetooth dongle).

[0032] In step 380, the autorun firmware loads the firmware associated with the enumerated USB peripheral (e.g., USB flash drive or USB Bluetooth dongle).

[0033] In step 390, the autorun firmware is configured to not re-enumerate itself and continues to act as a virtual CD-ROM type device implementing the MMC-2 specifications.

[0034] Process steps 320, 330, 340, 350 and 360 correspond to the autorun firmware implementation. Step 390 provides for the implementation of a virtual mass storage device from a memory component that implements SCSI command set and MMC-2 specifications.

[0035] Autorun firmware according to the present invention can be embedded into multiple USB device configurations to provide a variety of unique USB peripherals with autorun functionality and into other peripheral devices with similar functionality. For example, Fig. 4A shows a USB hub 400 in which a USB microcontroller 410 with auto run firmware 415 communicates with an internal memory component 420. In Fig. 4B, a USB microcontroller 450 is connected to an external memory component 460 through a USB downstream port 470.

[0036] With reference to Fig. 4A, the USB microcontroller 410 that forms a part of the USB hub 400 typically is a repeater type entity allowing for cascaded multiple USB peripherals to connect through a single upstream port to a host system. The USB microcontroller 410 includes support for programming capability, which includes the autorun firmware 415. The Autorun firmware can then be ported to work on the USB microcontroller 410. The firmware may be stored on the internal memory component 420. Alternatively, the Autorun firmware may be stored on external memory that is in an attached USB memory component 430.

[0037] As another configuration, Fig. 5 is a block diagram of a USB peripheral 500 having multiple functionalities. In this implementation, USB peripheral 500 includes an internal microprocessor with USB interfacing 510, or alternatively a USB microcontroller, that communicates with a memory component 520 and wireless (e.g., Bluetooth) networking hardware 530. As a result, USB peripheral 500

is capable of operating as a wireless (e.g., Bluetooth) networking device or "dongle" and as USB flash drive, both of which are accessible with autorun functionality

[0038] In one configuration, the microprocessor 510 has USB interfacing ability. It is coupled with a memory component 520 and Bluetooth radio component 530. Microprocessor 510 implements client layers of the Bluetooth stack. The firmware that the microprocessor 510 executes is stored in memory component 520. The autorun firmware can also be additionally stored as a part of the functionality of existing firmware or separately in the memory component 520. In another configuration, the microprocessor 510 may not directly have USB interfacing capability and could use a separate USB microcontroller (not shown).

[0039] A feature of including autorun firmware in USB peripherals is that software applications can be bundled with the USB peripherals. The bundled software application may or may not utilize the USB peripheral.

[0040] As an example, Fig. 6 is a block diagram of a USB hub 600 that includes a USB microcontroller 610 with autorun firmware 615 and access to one or multiple distinct functionalities or USB peripherals, such as an external memory component 630, a Bluetooth networking component 640, or a WLAN component 650. Such USB peripherals 630-650 could be formed in combination with USB hub 600. USB hub 600 may be externally connected with one or more of these components 630-650, as illustrated, or alternatively one or more of the components 630-650 can be internally integrated to form a USB peripheral or device with multiple distinct functionalities.

[0041] There could be multiple executions of autorun firmware from each or some of these peripherals. Thus the autorun firmware allows for distribution of software (e.g. device drivers, synchronization software, etc.) that can be autorun along with any USB peripheral.

[0042] The implementation options also include mechanisms for allowing the autorun feature to be enabled or disabled by an external

mechanism (e.g., switch) that is included on the device or peripheral. The switch could be manually operable by a person. The switch could be a simple two-mode (e.g., autorun on/off) switch or could be a switch that selects from among more than two modes.

[0043] Fig. 7 is a schematic diagram of a person-operable physical slide switch 700 that allows a person to select from among multiple modes, functionalities, or peripherals available on a USB device or "dongle." As an example, switch 700 relates to features or peripherals available from USB hub 600 of Fig. 6, including external memory component 630 , and wireless dongle or module (640 or 650) for adding wireless (e.g. Bluetooth, WiFi, IrDA) interface to its host PC.

[0044] In this exemplary illustration, switch 700 has 4 user-selectable positions. In position 710, autorun functionality is enabled, the wireless component is disabled. In position 720, autorun functionality is disabled, wireless component is disabled. In position 730, autorun functionality is enabled, wireless component is enabled. In position 740, autorun functionality is disabled, wireless component is enabled.

[0045] The autorun firmware enables the distribution of software that can be autorun from a memory component. There is also a unique security mechanism that can be incorporated to protect the software that is installable or executable from the memory component by the autorun firmware.

[0046] A section of the internal memory component (e.g., memory component 620, Fig. 6) may be protected from public access by password protecting it or by physical security means such as a lock, among other means. The flash memory component can also be segmented into public and private sections. Private sections can be used to store installable or executables that cannot be viewed or accessed by the user, and public sections can be viewed or accessed by users in a conventional manner. The installable or executable software being distributed through the memory component can be stored in the protected region of the memory component. Security by way of copy protection of this installable software can be achieved by

allowing only an application launcher executable, which is autorun from the memory component, to access the installable software.

[0047] In one implementation, the application launcher executable has the following characteristics: it is autorun from memory component, and it has access to the protected or private region of memory component. This access is gained by authenticating itself to the memory controller (e.g. USB microcontroller) and/or to the installable software in the protected region of the memory component. The authentication mechanism may be a password-based mechanism or a more involved cryptographic algorithm. Among the various techniques used for authentication are digital signatures and unique identifiers like the Bluetooth Device Address, MAC address, etc. The application launcher executable may authenticate itself directly to the memory controller software and/or installable software or to a separate authentication software that resides in the protected region of the memory component.

[0048] The application launcher executable may be built generically to execute any or all executables and installables that exist within the protected region of the memory component. Alternatively, the application launcher executable may be programmed to launch a particular executable or installable from the protected region. Considering the possibility of the memory component being segmented into "n" protected sections where n is greater than 1, the application launcher executable may access one or more of these sections in the mechanism described herein. The protected memory region may contain, for example, executable software (also called an application executable), or installable software (also called an application installable), or protected data.

[0049] Fig. 8 is a flow diagram of a software-implemented copy protection method for protecting of software that is executable or installable on using autorun firmware.

[0050] In step 810, an application launcher executable that is stored in a memory component of an IC memory device is run automatically on a

host computer by an autorun firmware stored on the IC memory device. The autorun firmware is operates automatically upon activation of the IC memory device, such as occurs when the memory device is plugged into a port or socket of the host computer.

[0051] In step 820, the application launcher authenticates itself to authentication agent software that resides in the protected region of the memory component. The authentication agent software may be incorporated within the software executable or installable that is being protected or may be a separate application. The authentication algorithm may be password based or may involve cryptographic techniques.

[0052] Step 830 is a query whether the authentication is successful. If not, access to the protected executable or installable is denied. If authentication is successful, step 830 proceeds to step 840 and the application launcher executable gains access to the protected memory region.

[0053] In step 840, the application launcher executable executes the application executable or installable that is stored in the protected region of the memory component. The application launcher executable may also be programmed to execute any or all executables and installables that exist within the protected region of the memory component.

[0054] In step 850, the executables and installables thus launched are executed on the host computer.

[0055] In view of the many possible embodiments to which the principles of our invention may be applied, it should be recognized that the detailed embodiments are illustrative only and should not be taken as limiting the scope of our invention. Rather, we claim as our invention all such embodiments as may come within the scope and spirit of the following claims and equivalents thereto.